Free with CU Amiga Magazine December 1995 All rights reserved

FREE 132 PAGE AMIGA E GUIDE

Amiga E

The new Amiga programming language

CU Amiga Magazine

Amiga E

○ ○ ○ ○

Amiga E

A new and fast programming language for your Amiga

[illegible]

Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 1

1. Introduction to Amiga E

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

1.1 A Simple Program

[illegible]

1.1.2 The code

Enter the following [illegible] sample [illegible] press the RETURN or ENTER key [illegible]

```
PROC main()
  WriteF('My first prog[illegible]
ENDPROC
```

[illegible]

1.1.3 Compilation

[illegible]

```
ec simple
```

[illegible]

1.1.4 Execution

On [illegible] entering the [illegible]

```
simple
```

[illegible]

```
1 Ram Disk [illegible]
Amiga E Compil[illegible]
registered [illegible]
analysing [illegible] compil[illegible]
no errors
1 Ram Disk [illegible]
My first program [illegible]
```

[illegible]

Chapter 2

2. Understanding a Simple Program

[illegible]

2.1 Changing the Message

[illegible]

2.1.1 Tinkering with the example

[illegible]

2.1.2 Brief overview

[illegible]

Procedures [illegible]

Parameters [illegible]

[illegible]

2.2 Procedures

[illegible]

2.2.1 Procedure Definition

[illegible] ENDPROC [illegible]

2.2.2 Procedure Execution

[illegible]

2.2.3 Extending the example

[illegible]

```
PROC main()
  WriteF( My first program
  fred()
ENDPROC

PROC fred()
  WriteF(     slightly improve
ENDPROC
```

[illegible]

2.3 Parameters

[illegible]

[illegible]

2.4 Strings

[illegible]

```
  PROC main()
    WriteF('My fir[illegible]
    fred()
  ENDPROC

  PROC fred()
    WriteF([illegible]
  ENDPROC
```

[illegible]

2.5 Style, Reuse and Readability

[illegible]

```
  PROC main()
    WriteF('My first program\n')
    WriteF('[illegible]slightly improved\n')
  ENDPROC
```

[illegible]

[illegible]

2.6 The Simple Program

[illegible]

Chapter 3

3 Variables and Expressions

[illegible]

3.1 Variables

[illegible]

3 1 1 Variable types

[illegible]

[illegible]

3.1.2 Variable declaration

[illegible]

```
DEF x
DEF a  b  c
```

[illegible]

3.1.3 Assignment

[illegible]

```
x = 2
```

[illegible]

```
x   x
```

[illegible]

```
x   x
```

3.1.4 Global and local variables (and procedure parameters)

[illegible]

```
DEF w

PROC main
DEF x
   x = 2
   w  1
   fred(
ENDPROC

PROC fred
DEF
   [illegible]
   x = 2
ENDPROC
```

[illegible]

```
DEF w

PROC main()
DEF x
   x :=2
   w :=1
   fred()
ENDPROC

PROC fred()
DEF x
   x :=3
   w :=2
ENDPROC
```

[illegible]

[illegible]

[illegible]

```
DEF y

PROC onemore(x)
   y :=x+1
ENDPROC
```

[illegible]

[illegible] onemore:

```
onemore(120)
onemore(52+34)
onemore(y)
```

A procedure can be defined [illegible] number of parameters. [illegible]

```
DEF y

PROC addthem(a, b)
   a :=a+2
   y :=a*b
ENDPROC
```

The following are some examples of calling addthem:

```
addthem(120,-20)
addthem(y,y+1)
```

[illegible]

3.1.5 Changing the exemple

[illegible]

[illegible]

```
PROC main()
  WriteF( My first program\n )
  fred()
ENDPROC

PROC fred()
  WriteF(    brought to you by the
  number \d\n   236)
ENDPROC
```

[illegible]

```
PROC main()
  WriteF( My first program\n )
  fred()
ENDPROC

PROC fred()
  WriteF(    brought to you by the number
  \d and \d\n   16  236)
ENDPROC
```

[illegible]

[illegible]

```
PROC main()
  WriteF( My first program\n )
  fred(16  236)
ENDPROC
```

```
PROC fred([illegible])
  WriteF(   [illegible]
  numbe [illegible]
ENDPROC
```

[illegible]

3.2 Expressions

[illegible]

3.2.1 Mathematics

[illegible]

```
1+2+3+4
15 5
330/33
3*3+1
```

[illegible]

[illegible]

[illegible]

3.2 2 Logic and comparison

[illegible]

[illegible]

```
[illegible]
[illegible] FALSE
[illegible]
[illegible] AND FALSE
[illegible]
```

[illegible]

[illegible]

3.2.3 Precedence and grouping

[illegible]

[illegible]

[illegible]

[illegible]

```
[illegible]
```

[illegible]

```
[illegible]
```

[illegible]

```
[illegible]
```

[illegible]

Chapter 4

4. Program Flow Control

[illegible]

4.1 Conditional Block

[illegible]

```
IF x>0
   x:=x+1
   WriteF('Increment: x is now \d\n', x)
ELSEIF x<0
   x:=x-1
   WriteF('Decrement: x is now \d\n', x)
ELSE
   WriteF('Zero: x is 0\n')
ENDIF
```

[illegible]

[illegible]

```
SELECT x
  CASE 0
    WriteF('x is zero\n')
  CASE 1
    WriteF('x is one\n')
  CASE 2
    WriteF('x is 2\n')
  DEFAULT
    WriteF('x is not zero, one or 2\n')
ENDSELECT
```

[illegible]

[illegible]

4.1.1 'IF' block

The IF block [illegible] SION [illegible]

```
IF <EXPRESSIONA>
   <STATEMENTSA>
ELSEIF <EXPRESSIONB>
   <STATEMENTSB>
ELSE
   <STATEMENTSC>
ENDIF
```

[illegible]

- [illegible]
- [illegible]
- [illegible]

[illegible]

```
IF <EXPRESSION> THEN
   <STATEMENTS>
```

[illegible] THEN keyword [illegible] ATTENTION [illegible] the IF block

```
IF x>0 THEN x:=x+1 ELSE x:=0
IF x>0
  x:=x+1
ELSE
  x:=0
ENDIF

IF x=0 THEN WriteF('x is zero\n')
IF x=0
  WriteF('x is zero\n')
ENDIF

IF x<0
  WriteF('Negative x\n')
ELSEIF x>2000
  WriteF('Too big x\n')
ELSEIF (x=2000) OR (x
[illegible]
ENDIF

[illegible]

[illegible]
```

[illegible]

```
IF [illegible]
  [illegible]
[illegible]
  [illegible]
[illegible]
  [illegible]
EN[illegible]

IF x>0
   Write[illegible]
ELSE
   Write[illegible]
ENDIF
```

[illegible]

[illegible] The whole IF block [illegible] ELSEIF [illegible]

4 1 2 'IF' expression

[illegible] IF block [illegible]

```
IF x>0
   y  x+1
ELSE
   y -0
ENDIF
```

[illegible]

```
y =(IF x>0 THEN x+1 ELSE 0)
```

The parentheses are [illegible] more readable. [illegible]

[illegible] expression has the following syntax

```
IF  EXP> THEN <EXPA> ELSE <EXPB>
```

[illegible] expressions [illegible] ELSEIF [illegible]

[illegible]

4 1 3 'SELECT' block

The SELECT block has the [illegible]

```
SELECT <VARIABLE>
```

[illegible]

4.1 4 'SELECT OF' block

The SELECT OF block [illegible] than the normal SELECT block [illegible] the following form:

```
SELECT <MAXRANGE> OF <EXPRESSION>
CASE <CONSTA>
   <STATEMENTSA>
CASE <CONSTB1> TO <CONSTB2>
   <STATEMENTSB>
CASE <RANGE1> <RANGE2>
   <STATEMENTSC>
DEFAULT
   <STATEMENTSD>
ENDSELECT
```

[illegible] EXPRESSION [illegible] the normal SELECT [illegible] MAXRANGE [illegible] CONSTA [illegible] CONSTB1 [illegible] CONSTB [illegible] MAXRANGE [illegible] MAXRANGE [illegible]

[illegible] CASE [illegible] CONSTB [illegible] CONSTB1 [illegible] CASE [illegible]

[illegible] 4 5

[illegible] TO 3 4 TO 5

[illegible] EXPRESSION [illegible] MAXRANGE [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

4.2 Loops

[illegible]

4.2.1 'FOR' loop

[illegible]

```
PROC main()
  DEF x
FOR x := 1 TO 100
  WriteF('\d ', x)
ENDFOR
  WriteF('\n')
ENDPROC
```

[illegible] ENDFOR [illegible]

[illegible]

```
    WriteF( [illegible]
    WriteF( [illegible]

    [illegible]
    [illegible]
  [illegible]
```

[illegible] FOR loop [illegible]

```
  F <VAR> [illegible] EXPRESSIONA> TO [illegible]SSIONB>
  TEP <NUMBER>
    <STATEMENTS>
  ENDFOR
```

The VAR [illegible] EXPRESSIONA [illegible] EXPRESSIONB [illegible] The STEP [illegible] NUMBER [illegible] STEP [illegible]

[illegible]

```
  [illegible] VAR> [illegible] <EXPA> TO <EXPB> STEP <EXPC
  [illegible] STATEMENT>
```

4.2.2 'WHILE' loop

The FOR [illegible] WHILE [illegible]

```
  PROC main()
    DEF x
      x:=1
    WHILE x<=100
      WriteF( \d [illegible] x)
      x :=x+1
    ENDWHILE
      WriteF( \n )
    ENDPROC
```

We've replaced the FOR [illegible] WHILE loop [illegible] the FOR [illegible] FOR loop [illegible]

[illegible] ENDWHILE [illegible]

```
  PROC main()
  DEF x y
    x [illegible]
    y [illegible]2
  WHILE (x<10) AND (y<10
    WriteF( x is \d and y is [illegible]d\n [illegible] x [illegible] y
    x [illegible] x+2
    y [illegible]y+2
  ENDWHILE
  ENDPROC
```

[illegible] WHILE [illegible]

[illegible] FOR [illegible]
WHILE [illegible]

```
[illegible] <EXPRESSION> DO <STATEMENT>
```

[illegible] was a big problem. FOR [illegible] reach the limit [illegible] However WHILE [illegible] forever and [illegible] provide [illegible] loop [illegible] the END [illegible]

4 2 3 'REPEAT UNTIL' loop

A REPEAT UNTIL loop [illegible] WHILE loop [illegible] where [illegible] back [illegible] program [illegible] REPEAT UNTIL loop [illegible]

```
PROC main()
   DEF x
   x  1
REPEAT
   WriteF(  d     x)
   x  x+1
UNTIL x>100
   Wr  eF    n )
ENDPROC
```

[illegible] WHILE [illegible] UNTIL [illegible] REPEAT UNTIL [illegible] WHILE [illegible]

[illegible] UNTIL [illegible] REPEAT UNTIL [illegible]

[illegible] middle [illegible] printed [illegible]

Chapter 5

5 Summary

[illegible]

[illegible]

```
 1  PROC main
 2     DEF x y
 3     x  1
 4     y =2
 5  WHILE (x<10) AND  y<1[illegible]
 6     Write [illegible] x is  d and y is  d [illegible]  x
 7     x  x 2
 8     y  y 2
 9     ENDWHILE
10  ENDPROC
```

[illegible]

LINE(S)	OBSERVATION
1-10	The procedure definition
1	The declaration of the procedure **main** with no parameters.
2	The declaration of local variables 'x' and 'y'
3, 4	Initialisation of 'x' and 'y' using assignment statements
5-9	The 'WHILE' loop
5	The loop check for the 'WHILE' loop using the logical operator 'AND' the comparison operator '<' and parentheses to group the expression
6	The call to the (built-in) procedure 'WriteF' using parameters. Notice the string, the place holders for numbers, '\d' and the linefeed, '\n'
7, 8	Assignments to 'x' and 'y' adding two to their values.
9	The marker for the end of the 'WHILE' loop
10	The marker for the end of the procedure

Chapter 6

6. Procedures and Functions

[illegible]

6.1 Functions

[illegible]

```
PROC main()
DEF sum
   sum:=12+79
   WriteF('Using +, sum is \d\n', sum)
   sum:=add(12,79)
   WriteF('Using add, sum is \d\n', sum)
ENDPROC

PROC add(x, y)
DEF s
   s:=x+y
ENDPROC s
```

[illegible]

```
Using +, sum is 91
Using add, sum is 91
```

[illegible]

[illegible]

```
PROC add(x y
DEF s
  s =x+y
RETURN s
ENDPROC
```

[illegible] RETURN [illegible]

```
PROC limitedadd x y
  IF x>10000
RETURN 10000
  ELSEIF x< [illegible]000
RETURN  10000
  ELSE
RETURN x+y
ENDIF
 * The following code [illegible] *
x  )
IF x [illegible] THEN RETURN 9999 EL[illegible] R[illegible]RN  9999
ENDPROC
```

[illegible] RETURN [illegible]

[illegible] ENDPROC [illegible] RETURN [illegible]

6.2 One-Line Functions

[illegible] FOR [illegible]

```
PROC <NAME> [<ARG1>  <ARG2>          IS
   <EXPRESSION>
```

[illegible] RETURN [illegible]

```
PROC <NAME> [<ARG1>  <ARG2>          RETURN
   <EXPRESSION>
```

[illegible]

```
PROC add(x y  IS x+y
```

6.3 Default Arguments

[illegible]

```
PROC play(track
  Write( playing to play track  d\
  track
 * Rest of the code    */
ENDPROC
```

```
PROC main()
  play(1)       -> Start playing from track 1
  play(6)       -> Start playing from track 6
  play()        -> Start playing from track 1
ENDPROC
```

[illegible]

[illegible]

[illegible]

```
PROC fred(x, y, z) IS x+y+z  -> No
defaults

PROC fred(x, y, z=1) IS x+y+z  -> z
defaults to 1

PROC fred(x, y=23, z=1) IS x+y+z  -> y
and z have defaults
```

[illegible]

```
PROC fred(x, y=23, z) IS x+y+z  -> 
Illegal: no z default
```

```
PROC fred(x=9, y, z=1) IS x+y+z  -> 
Illegal: no y default
```

[illegible]

[illegible]

```
PROC fred(x, y=23, z=1)
  WriteF('x is \d, y is \d, z is \d\n',
  x, y, z)
ENDPROC

PROC main()
  fred(2, 3, 4)  -> No defaults used
  fred(2)  -> y and z default
  fred()  -> Illegal: x has no default
ENDPROC
```

[illegible]

```
  fred(2, 23, 9)  -> Need to supply 23 for y
```

[illegible]

6.4 Multiple Return Values

[illegible]

[illegible] ENDPROC, RETURN [illegible]

```
PROC movediag(x, y) IS x+8, y+4
```

[illegible]

```
PROC main()
   DEF a, b
   a, b :=movediag(10, 3)
   /* Now a should be 10+8, and b should >
   be 3+4 */
   WriteF('a is \d, b is \d\n', a, b)
ENDPROC
```

[illegible]

```
a, b:=6+movediag(10,3)  /* No obvious
value for b */
```

[illegible]

```
PROC main()
   DEF a, b
      /* The next two lines ignore the >
      second return value */
      a:=movediag(10, 3)
      WriteF('x coord of movediag(21, 4) >
      is \d\n', movediag(21,4))
ENDPROC
```

Chapter 7

7. Constants

[illegible]

7.1 Numeric Constants

[illegible]

[illegible]

[illegible] ASCII [illegible]

[illegible]

NUMBER	DECIMAL VALUE
21	21
[illegible]	143
[illegible]	26
$B1	177
%1110	14
%1010	10
[illegible]	122
Je	19 045
-A	-65

7.2 String Constants: Special Character Sequences

[illegible]

SEQUENCE	MEANING
\0	A null (ASCII zero)
\a	An apostrophe
\b	A carriage return (ASCII 13)
\e	An escape (ASCII 27)
\n	A linefeed (ASCII 10)
\q	A double quote (ASCII 34)
\t	A tab (ASCII 9)
\\	A backslash \

[illegible]

```
WriteF( Here  s an apostrophe  n
```

7.3 Named Constants

[illegible] CONST [illegible]

```
CONST ONE=1  LINEFEED=10  BIG NUM=999999
```

[illegible] ONE [illegible] BIG NUM 999999 [illegible]

[illegible] CONST [illegible]

```
CONST ZERO 0
CONST ONE ZERO+1
```

[illegible]

7.4 Enumerations

[illegible] would use an enumeration like this:

```
ENUM LONDON  MOSCOW  NEW YORK  PARIS  ROME
TOKYO
```

The ENUM keyword begins the [illegible] CONST keyword [illegible]

```
CONST LONDON 0  MOSCOW=1  NEW YORK=2
PARIS 3  ROME 4  TOKYO=5
```

[illegible]

[illegible]

```
ENUM APPLE ORANGE CAT=55 DOG GOLDFISH

CONST APPLE 0 ORANGE 1 CAT 55 DOG 56 >
GOLDFISH 57
```

7.5 Sets

[illegible]

```
SET ENGLISH FRENCH GERMAN JAPANESE >
RUSSIAN

CONST ENGLISH 1 FRENCH 2 GERMAN 4 >
JAPANESE 8 RUSSIAN 16
```

[illegible]

```
CONST ENGLISH %00001 FRENCH %00010 >
GERMAN %00100 JAPANESE=%01000 >
RUSSIAN=%10000
```

[illegible] ENGLISH [illegible] OR [illegible] ENGLISH [illegible] JAPANESE [illegible] ENGLISH and JAPANESE [illegible] AND [illegible] FRENCH [illegible] AND and OR [illegible]

```
[illegible] GERMAN [illegible] RUSSIAN
[illegible] speak [illegible] there
[illegible] speak AND JAPANESE
[illegible] speak [illegible]
[illegible]
[illegible]
ENDIF
[illegible] AND [illegible]
[illegible]
[illegible]
ELSE
[illegible]
[illegible]
ENDIF
```

[illegible]

Chapter 8

8. Types

[illegible]

[illegible]

[illegible]

8.1 'LONG' Type

[illegible]

8.1.1 Default type

[illegible]

[illegible]

```
[illegible] ( [illegible] p LONG  q  r LONG)
[illegible] LONG
    <STATEMENTS>
[illegible] PROC
```

[illegible]

8 1 2 Memory addrasses

[illegible]

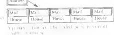
8.2 'PTR' Type

[illegible]

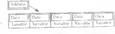
8 2.1 Addresses

[illegible]

[illegible]



[illegible]



8 2.2 Pointers

[illegible]

```
[illegible] :INT
    [illegible] PTR TO gadget
```

[illegible]

8 2 3 Indirect types

[illegible]

```
/* This program fragment [illegible] legal
declarations */
DEF c CHAR [illegible] INT
/* This program fragment [illegible] legal
declarations */
PROC fred(a INT  b CHAR)
DEF x INT
   <STATEMENTS>
ENDPROC
```

[illegible] INT [illegible] CHAR [illegible] LONG [illegible] LONG [illegible] CHAR [illegible]

8.2 4 Finding addresses (making pointers)

[illegible]

```
DEF x

PROC main()
   fred(2)
ENDPROC

PROC fred(y)
   DEF z
      WriteF('x is at address \d\n', {x})
      WriteF('y is at address \d\n', {y})
      WriteF('z is at address \d\n', {z})
ENDPROC
```

[illegible]

8.2 5 Extracting data (dereferencing pointers)

[illegible]

[illegible]

```
PROC main()
  fred
ENDPROC

PROC fred
  DEF x, p:PTR TO LONG
  x:=1
  p:={x}
  barney(p)
  WriteF('x is now \d\n', x)
ENDPROC

PROC barney(ptr:PTR TO LONG)
  DEF val
  val:=^ptr
  ^ptr:=val+6
ENDPROC
```

[illegible]

```
PROC main()
  fred
ENDPROC

PROC fred()
  DEF x [illegible]
  x:=1
  [illegible]
  [illegible]
  x[illegible]
  WriteF([illegible])
ENDPROC
```

[illegible]

```
PROC main
  fred
ENDPROC

PROC fred
  DEF x
  x:=3
  barney({x})
  WriteF('x [illegible]', x)
ENDPROC

PROC barney(ptr)
  ^ptr:=[illegible]
ENDPROC
```

[illegible]

[illegible]

[illegible]

8.2.6 Procedure parameters

[illegible] PERTOINT [illegible]

8.3 'ARRAY' Type

[illegible] ARRAY [illegible]

8.3.1 Tables of data

[illegible] ARRAY [illegible]

```
DEF a[132]:ARRAY,
    table[21]:ARRAY OF LONG,
    ints[3]:ARRAY OF INT,
    objs[54]:ARRAY OF myobject
```

[illegible]

[illegible]

[illegible]

8.3.2 Accessing array data

[illegible]

```
DEF a[10]:ARRAY

PROC main()
  DEF i
  FOR i:=0 TO 9
  a[i]:=i*i
 ENDFOR
  WriteF('The 7th element of the array a is \d\n', a[6])
  a[7]:=10
  WriteF('The array is now:\n')
  FOR i:=0 TO 9
  WriteF(' a[\d] = \d\n', i, a[i])
 ENDFOR
ENDPROC
```

[illegible]

[illegible]

8.3.3 Array pointers

[illegible]

```
DEF a[10] ARRAY OF INT

PROC main()
   DEF ptr PTR TO INT  i
   FOR i -0 TO 9
      a[i] -i
ENDFOR
   ptr  a
   ptr++
   ptr[]  22
FOR i =0 TO 9
   WriteF( a[\d] is \d\n   i  a[i])
ENDFOR
ENDPROC
```

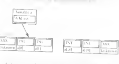
[illegible]

[illegible]

```
DEF a[2] ARRAY OF INT

DEF a PTR TO INT
```

[illegible]



[illegible]

8.3.4 Point to other elements

[illegible]

```
addr  p
```

```
[illegible]
```

[illegible]

8 3 5 Array procedure parameters

[illegible]

```
DEF a[10]:ARRAY OF INT

PROC main()
   DEF i
   fillin(a,10)
   FOR i:=0 TO 9
WriteF('a[\d] is \d\n',i,a[i])
   ENDFOR
ENDPROC
```

```
PROC fillin(ptr:PTR TO INT,x)
   DEF i
   FOR i:=0 TO x-1
   ptr[]:=i
   ptr++
 ENDFOR
   ENDPROC
```

[illegible]

```
PROC fillin2(ptr:PTR TO INT,x)
   DEF i
   FOR i:=0 TO x-1
   ptr[i]:=i
   ENDFOR
 ENDPROC
```

[illegible]

```
PROC fillin3(ptr:PTR TO INT,x)
   DEF i
   FOR i:=0 TO x-1 DO ptr[]++:=i
 ENDPROC
```

8.4 'OBJECT' Type

[illegible]

8.4 1 Example object

[illegible]

[illegible]

```
OBJECT rec
   tag  check
   table[8] ARRAY
   data LONG
ENDOBJECT

PROC main()
   DEF a rec
   a tag  1
   a check  a
   a data  a tag+10000*a  x9
ENDPROC
```

[illegible]

[illegible]

8 4 2 Element selection and element types

[illegible]

[illegible]

[illegible]

```
  re
   tag  che k
   table[8  ARRAY
   data LONG
ENDOBJECT

OBJECT bigre
   data PT  TO LONG
   check PTR TO re
   [illegible]
ENDOBJECT

PROC main
   DEF  re    bigre     t PTR    re
   a  al e [   ]
   b sub  c
   b  ubl    s
      sub    al      [
   b re  ab       able [1  A
   rt [  e  al e
    t[  c  b
   rt[  tab e       a
ENDPROC
```

[illegible]

```
b subrec tag
```

[illegible]

```
b rec able[0  able[b]  A
```

[illegible]

[illegible]

```
    [illegible]
      [illegible]
      t-[illegible]
      data [illegible]
  ENDOBJE[illegible]

  PROC main
    DEF a[10 [illegible]
    p -=
    FOR i -0 [illegible]
    a[i] tag [illegible]
    p check[illegible]
  ENDFOR
  FOR i -0 TO 9
    IF a[i] tag<>a[i  check
    WriteF( Whoops  a[\d] went [illegible]
  ENDIF
    ENDFOR
  ENDPROC
```

It [illegible] the same as [illegible] first element of the [illegible] the [illegible] [illegible]

8.4 3 Amiga systam objects

[illegible]

[illegible] the [illegible]

8.5 'LIST' and 'STRING' Types

[illegible] STRING [illegible] ARRAY [illegible] CHAR [illegible] LONG [illegible]

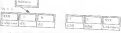
8 5.1 Normal strings and E-strings

[illegible] CHAR [illegible]

```
  DEF s PTR TO CHAR
  s  This is a string constant
  /* Now s[] is " and s[2] is   *
```

[illegible] LONG [illegible]

```
  DEF s
  s = This is a string constant
```



[illegible] the last

[illegible]

```
DEF s[30]:STRING
```

[illegible]

8.5.2 String functions

[illegible]

```
DEF s[37]:STRING

DEF s:PTR TO CHAR
s:=String(37)
```

[illegible]

StrCmp(STRING [illegible]

[illegible]

StrCopy(STRING [illegible] LENGTH> ALL [illegible]

[illegible]

```
DEF s[ ]:STRING
StrCopy(s, 'More [illegible]
```

[illegible]

```
/* You don't want to do things like this */
DEF s[80]:STRING
s:='This is a string constant'
```

[illegible]

```
DEF s[80]:STRING
StrCopy(s,'This is a string constant')
```

[illegible]

StrAdd(<E-STRING>,<STRING>,<LENGTH>=ALL)

[illegible]

```
DEF s[30]:STRING
StrCopy(s,'This is a string',ALL)
StrAdd(s,' and a half')
```

StrLen(<STRING>)

[illegible]

```
StrLen('abc')
StrLen('abc\0def')
```

[illegible]

EstrLen(<E-STRING>)

Returns the length of E-STRING [illegible]

StrMax(<E-STRING>)

Returns the maximum length of E-STRING [illegible]

RightStr(<E-STRING1>,<E-STRING2>,<LENGTH>)

[illegible]

MidStr(<E-STRING>,<STRING>,<INDEX>,<LENGTH>=ALL)

[illegible]

```
DEF s[30]:STRING
MidStr(s,'Just four',5)
MidStr(s,'Just four apples',5,4)
```

InStr(<STRING1>,<STRING2>,<STARTINDEX>=0)

[illegible] STRING [illegible] STRING [illegible] STARTINDEX [illegible] STRING [illegible] STARTINDEX [illegible] STRING [illegible]

TrimStr(<STRING>s)

[illegible]

```
DEF s:PTR TO CHAR
s:=TrimStr( [illegible] \n [illegible] t [illegible] 2345
```

LowerStr(<STRING>s)

[illegible] Upper [illegible]

UpperStr(<STRING>s)

[illegible]

SetStr(<E-STRING>,<LENGTH>)

[illegible] STRING [illegible] LENGTH [illegible]

```
DEF s[10]:STRING, s[0] [illegible]
/* Remember that [illegible] is a [illegible] value */
[illegible]
/* At this point s is just an
array of CHAR */
SetStr(s, [illegible] 3)
/* Now s can be used as an E-string
again */
SetStr(s, [illegible])
/* s is a bit shorter [illegible]
string */
```

[illegible]

Val(<STRING>,<ADDRESS>=NIL)

[illegible] STRING [illegible] ADDRESS [illegible] ADDRESS [illegible] ADDRESS [illegible]

```
DEF s[10]:STRING, value, [illegible] p:PTR
TO CHAR
StrCopy(s, [illegible] $3F [illegible] 0 [illegible]
value, chars:=Val( [illegible] 0 [illegible]
return values
/* After the above line value and chars
will both be zero */
value:=Val(s, {chars}) [illegible] address
of chars
/* value is now 10, chars will be [illegible] */
p:=s+chars
/* p now points to the space after the 0
in s */
value, chars:=Val(p)
/* Now value is $3F (63), chars is 6 */
```

[illegible]

8.5.3 Lists and E-lists

[illegible]

```
DEF list:PTR TO LONG,number
number:=22
list:=[1,2,number]
```

[illegible]

```
DEF list[3]:ARRAY OF LONG,number
number:=22
list[0]:=1
list[1]:=2
list[2]:=number
```

[illegible]

```
DEF lt[30]:LIST
```

[illegible]

[illegible]

8.5.4 List functions

[illegible]

List(MAXSIZE)

[illegible]

```
DEF lt[46]:LIST
```

```
DEF lt:PTR TO LONG
lt:=List(46)
```

[illegible]

ListCmp(LIST1,LIST2,LENGTH=ALL)

[illegible]

```
ListCmp([1,2,3],[1,2,3,1])
ListCmp([1,2,3,4],[1,2,3,7],3)
```

ListCopy(ELIST,LIST,LENGTH=ALL)

[illegible]

```
DEF lt[7]:LIST,x
x:=4
ListCopy(lt,[1,2,3,x])
```

[illegible]

ListAdd(<E-LIST>,<LIST>,<LENGTH>=ALL)

Works just like StrAdd() [illegible] the E-list [illegible]

```
DEF lt[30]:LIST
ListCopy(lt,[1,2,3,4])
ListAdd(lt,[5,6,7,8])
```

ListLen(<LIST>)

Works just like StrLen() [illegible] LIST [illegible]

ListMax(<E-LIST>)

Works just like StrMax() [illegible] E-LIST

SetList(<E-LIST>,<LENGTH>)

Works just like SetStr() [illegible] LIST [illegible] LENGTH

ListItem(<LIST>,<INDEX>)

Returns the element [illegible] LIST [illegible] INDEX [illegible]

```
WriteF(ListItem(['Fred','Barney','Wilma'],
name))

DEF lt:PTR TO LONG
lt:=['Fred','Barney','Wilma']
WriteF(lt[name])
```

8.5.5 Complex types

[illegible] STRING [illegible] LIST [illegible]

8.5.6 Typed lists

[illegible] LONG [illegible] CHAR, INT [illegible]

```
DEF ints:PTR TO INT, objects:PTR TO rec,
p:PTR TO CHAR
ints:=[1,2,3]:INT
p:='fred'
objects:=[1,2,p,4,300,301,'barney',303]:rec
```

[illegible]

```
DEF ints[3]:ARRAY OF INT, objects[2]:ARRAY
OF rec, p:PTR TO CHAR
ints[0]:=1
ints[1]:=2
ints[2]:=3
p:='fred'
objects[0].tag:=1
objects[0].check:=2
objects[0].table:=p
objects[0].data:=4
objects[1].table:='barney'
objects[1].tag:=300
objects[1].data:=303
objects[1].check:=301
```

[illegible]

[illegible]

Chapter 9

9 More About Statements and Expressions

[illegible]

9.1 Initialised Declarations

[illegible]

```
SET ENGLISH  FRENCH  GERMAN  JAPANESE
RUSSIAN

CONST FREDLANGS=ENGLISH OR FRENCH OR GERMAN

DEF fredspeak FREDLANGS
   p=NIL PTR TO LONG  q=0 PTR TO rec

PROC fred()
   DEF x 1  y 88
   /* Rest of procedure */
ENDPROC
```

[illegible]

[illegible]

9.2 Assignments

[illegible]

```
IF (x:=y*z)=0
  WriteF('Error: y*z is zero (and x
  is zero)\n')
ELSE
  WriteF('OK: y*z is not zero (and x
  is y*z)\n')
ENDIF
```

```
x:=y*z
IF x=0
  WriteF('Error: y*z is zero (and x is
  zero)\n')
ELSE
  WriteF('OK: y*z is not zero (and x is
  y*z)\n')
ENDIF
```

[illegible]

```
VAR
  [illegible] EXPRESSION
  [illegible] ELEMENT NAME
[illegible]
```

[illegible]

```
x
x
x [illegible]
x [illegible]
x [illegible]
x [illegible]

x [illegible]
[illegible]
x [illegible]
```

[illegible]

9.3 More Expressions

[illegible]

9.3.1 Side-effects

[illegible]

[illegible]

9 3.2 'BUT' expression

[illegible]

```
fred((x:=12*3) BUT x+y)

x:=12*3
fred(x+y)
```

[illegible]

9 3 3 Bitwise 'AND' and 'OR'

[illegible]

X	Y	X OR Y	X AND Y
1	1	1	1
1	0	1	0
0	1	1	0
0	0	0	0

[illegible]

[illegible]

```
PROC main()
  test(TRUE    TRUE\t [illegible]
  test(FALSE   FALSE\t\
  test(1   \t\t
  test(4  4\t\t
   test(TRUE OR TRUE   TRUE OR TRUE\t )
   test(TRUE AND TRUE   TRUE AND TRUE\t
   test(1 OR 4   1 OR 4\t\t )
   test(1 AND 4  1 AND 4\t\t )
ENDPROC

PROC test(x  title)
   WriteF(title)
   WriteF(IF x THEN   is TRUE\n  ELSE   )
   is FALSE\n )
ENDPROC
```

[illegible]

[illegible]

9.3.4 'SIZEOF' expression

[illegible]

```
OBJECT rec
   tag  check
   table[8] ARRAY
   data LONG
ENDOBJECT

PROC main()
   WriteF( Size of rec object is \d )
   by es\n   SIZEOF rec)
ENDPRO
```

[illegible]

[illegible]

```
OBJECT rec2
   tag  check
   table[7] ARRAY
   data LONG
ENDOBJECT

PROC main()
WriteF( Size of rec2  bject is  d
bytes\n   SIZEOF rec2
ENDPROC
```

[illegible]

Chapter 10

10. E Built-In Constants, Variables and Functions

[illegible]

10.1 Built-In Constants

[illegible]

TRUE FALSE

[illegible] FALSE [illegible]

NIL

[illegible] NIL [illegible]

ALL

[illegible] ALL [illegible]

GADGETSIZE

[illegible]

OLDFILE, NEWFILE

[illegible]

STRLEN

[illegible]

```
PROC main()
  DEF s:PTR TO CHAR, len
  s:='12345678'
  len:=STRLEN
  WriteF(s)
  WriteF('\nis \d characters long\n', len)
ENDPROC
```

10.2 Built-In Variables

[illegible]

arg

[illegible]

```
fred file.txt "a big file" another
```

[illegible]

```
file.txt "a big file" another
```

[illegible]

wbmessage

[illegible]

[illegible]

stdin, stdout, conout

[illegible]

stdrast

[illegible]

dosbase, execbase, gfxbase, intuitionbase

[illegible]

10.3 Built-In Functions

[illegible]

10.3.1 Input and output functions

WriteF(<STRING>,<PARAM1>,<PARAM2>...)

[illegible]

PLACE-HOLDER	PARAMETER TYPE	PRINTS
[illegible]	Number	Character
\d	Number	Decimal number
\h	Number	Hexadecimal number
\s	String	String

[illegible]

```
PROC main()
   DEF s[30]:STRING
   StrCopy(s, 'Hello world - All!')
   WriteF('The third element of s is \
     [illegible] s[2])
   WriteF(' or \d (decimal)\n', s[2])
   WriteF(' or \h (hexadecimal)\n', s[2])
   WriteF(' and s itself is \s\s\s\n', s)
ENDPROC
```

[illegible]

SEQUENCE	MEANING
\l	Left justify in field
\r	Right justify in field
\z	Set fill character to "0"

[illegible]

```
PROC main()
   DEF s[30]:STRING
   StrCopy(s, [illegible] All')
   WriteF('The thi[illegible]
     \c \n', s[2])
   WriteF(' or \d[4] decimal[illegible] s[2]
   WriteF(' or \z\h[4] hexidecimal
  [illegible] s[2])
  WriteF('\s [illegible]
     elements of s [illegible]
  WriteF(' and s in a very [illegible]
     \s[20]\n' [illegible]
  WriteF(' and [illegible]
     [illegible]\s[20]\n' [illegible]
  ENDPROC
```

[illegible]

[illegible]

PrintF(<STRING>,<PARAM1>,<PARAM2>,...)

[illegible]

StringF(<ESTRING>,<STRING>,<ARG1>,<ARG2>,...)

[illegible]

```
DEF s[10]:STRING
StringF(s,'\s\d[5]','is a number:',123)
```

Out(<FILEHANDLE>,<CHAR>)

[illegible]

Inp(<FILEHANDLE>)

[illegible]

ReadStr(<FILEHANDLE>,<ESTRING>)

[illegible]

[illegible]

```
PROC main()
  DEF s[10]:STRING, [illegible]
  WriteF([illegible]
  [illegible] IF stdout [illegible]
  WHILE ReadStr([illegible]
    UpperStr(s)
  EXIT StrCmp(s,'QUIT',[illegible]
    WriteF('Read: [illegible]
  ENDWHILE
  WriteF([illegible]
ENDPROC
```

StrLength(<STRING>)

[illegible]

SetStdIn(<FILEHANDLE>)

[illegible] FILEHANDLE [illegible]

```
oldstdin=SetStdin newstdin

oldstdin=stdin
stdin=newstdin
```

SetStdOut(<FILEHANDLE>)

[illegible] FILEHANDLE [illegible]

10.3.2 Intuition support functions

[illegible]

[illegible]

OpenW(<X> <Y> <WID> <HGT> <IDCMP> <WFLGS> <TITLE> <SCRN> <SFLGS> <GADS> <TAGS>=NIL)

[illegible] NIL [illegible]

<X> <Y>

[illegible]

<WID> <HGT>

The width and height of the [illegible]

<IDCMP> <WFLGS>

The IDCMP [illegible]

<TITLE>

[illegible]

<SCRN> <SFLGS>

[illegible] SFLGS [illegible] SCRN [illegible] NIL [illegible] SFLGS [illegible] SCRN [illegible]

<GADS>

[illegible] NIL [illegible]

<TAGS>

[illegible] NIL [illegible]

[illegible] IDCMP [illegible]

IDCMP FLAG	VALUE
IDCMP NEWSIZE	$2
IDCMP REFRESHWINDOW	$4
IDCMP MOUSEBUTTONS	$8
IDCMP MOUSEMOVE	$10
IDCMP GADGETDOWN	$20
IDCMP GADGETUP	$40
IDCMP MENUPICK	$100
IDCMP CLOSEWINDOW	$200
IDCMP RAWKEY	$400
IDCMP DISKINSERTED	$8000
IDCMP DISKREMOVED	$10000

[illegible]

WINDOW FLAG	VALUE
WFLG SIZEGADGET	$1
WFLG DRAGBAR	$2
WFLG DEPTHGADGET	$4
WFLG CLOSEGADGET	$8
WFLG SIZEBRIGHT	$10
WFLG SIZEBBOTTOM	$20
WFLG SMART REFRESH	$0
WFLG SIMPLE REFRESH	$40
WFLG SUPER BITMAP	$80
WFLG BACKDROP	$100
WFLG REPORTMOUSE	$200
WFLG GIMMEZEROZERO	$400
WFLG BORDERLESS	$800
WFLG ACTIVATE	$1000

[illegible]

[illegible]

[illegible]

[illegible]

CloseW(<WINPTR>)

[illegible]

OpenS(<WID> <HGT> <DEPTH> <SCRNRES> <TITLE> <TAGS>) NBR

[illegible]

<WID> <HGT>

[illegible]

<DEPTH>

[illegible]

<SCRNRES>

[illegible]

<TITLE>

[illegible]

<FLAGS>

[illegible]

[illegible]

MODE FLAG	VALUE
V_LACE	$4
V_SUPERHIRES	$20
V_PFBA	$40
V_EXTRA_HALFBRITE	$80
V_DUALPF	$400
V_HAM	$800
V_HIRES	$8000

[illegible] V_HIRES [illegible] V_LACE [illegible] OR [illegible]

[illegible]

CloseS(<SCRNPTR>)

[illegible]

Gadget(<BUF> <GLIST> <ID> <FLAGS> <X> <Y> <WIDTH> <TEXT>)

[illegible]

<BUF>

[illegible]

<GLIST>

[illegible]

<ID>

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

BUTTON PRESSED	VALUE
Left	%001
Right	%010
Middle	%100

[illegible]

[illegible]

MouseX(<WINPTR>)

[illegible]

MouseY(<WINPTR>)

[illegible]

LeftMouse(<WINPTR>)

[illegible]

WaitMessage(<WINPTR>)

[illegible]

[illegible] MsgXXX [illegible]

```
PROC waitimessage(win:PTR TO window)
  DEF port,msg:PTR TO
  intuimessage,class,code,qual,iaddr
  port:=win.userport
  IF (msg:=GetMsg(port))=NIL
    REPEAT
      WaitPort(port)
    UNTIL (msg:=GetMsg(port))<>NIL
  ENDIF
  class:=msg.class
  code:=msg.code
  qual:=msg.qualifier
  iaddr:=msg.iaddress
  ReplyMsg(msg)
ENDPROC class
```

MsgCode()

[illegible] WaitIMessage.

MsgIaddr()

[illegible] WaitIMessage.

MsgQualifier()

[illegible] WaitIMessage.

WaitLeftMouse(<WINPTR>)

[illegible] WINPTR [illegible] IDCMP [illegible] IDCMP_MOUSEBUTTONS [illegible]

[illegible]

10.3.3 Graphics functions

[illegible]

[illegible]

Plot(<X>,<Y>,<PEN>=1)

[illegible]

Line(<X1>,<Y1>,<X2>,<Y2>,<PEN>=1)

[illegible] X1 Y1 [illegible]

Box(<X1>,<Y1>,<X2>,<Y2>,<PEN>=1)

[illegible] X1 Y1 [illegible]

Colour(<FORE PEN>,<BACK PEN>=0)

[illegible]

TextF(<X> <Y> <FORMAT STRING> <ARG1> <ARG2>

[illegible]

SetColour(<SCRNPTR> <PEN> <R> <G> <B>)

[illegible]

SetStdRast(<NEWRAST>)

[illegible]

```
oldstdrast = SetStdRast(newstdrast

oldstdrast  stdrast
stdrast  newstdrast
```

SetTopaz(<SIZE=8>)

[illegible]

10 3 4 Maths and logic functions

[illegible]

[illegible]

Abs(<EXPRESSION>)

[illegible]

Sign(<EXPRESSION>)

[illegible]

Even(<EXPRESSION>)

[illegible]

Odd(<EXPRESSION>)

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

RndQ(<EXPRESSION>)

Returns [illegible] EXPRESSION. [illegible]

Shl(<EXP1>,<EXP2>)

[illegible]

[illegible]

[illegible](<ADDR>) Int(<ADDR>) Char(<ADDR>)

[illegible]

[illegible]

PutLong(<ADDRE[illegible]
PutChar(<ADDRES[illegible]

[illegible]

10.3.5 System support functions

New(<BYTES>)

[illegible]

NewR(<BYTES>)

[illegible]

NewM(<BYTES>,<TYPE>)

[illegible]

```
MODULE [illegible]

PROC ma[illegible]
  DEF m
  m -NewM 20  MEMF [illegible]
  WriteF( Allocation [illegible]
  $\h\n   m)
EXCEPT
  IF exception  NEW" [illegible]
  WriteF( Failed\n )
ENDPROC
```

[illegible]ADDRESS>)

[illegible]

[illegible]OMPLEX>)

[illegible]

[illegible]

[illegible]

[illegible] BYTES>)

[illegible]

Wher [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Free with CU Amiga Magazine December 1995 All rights reserved

AMIGA

Amiga E guide • free with CU Amiga Magazine December 1995

FREE 132 PAGE AMIGA E GUID

Amiga E

The new Amiga programming language

CU Amiga Magazine